

HTML Template Engines

mediawiki.org/wiki/Requests_for_comment/HTML_templating_library

Why do we need one?

- MediaWiki lacks a standard MVC or similar framework
- Long term plan to move to HTML storage on the backend (users will need a new VE friendly template language)

We already have lots of use cases...

- **Flow** - uses a custom template system
- **ArticleCreationHelp** - uses a custom template system
- **PageTriage** - uses Underscore.js + backbone
- **MobileFrontend** - uses Hogan.js (Twitter's implementation of Mustache.js)
- **DonationInterface** - uses a custom template system (RapidHTML)
- **Fundraising thank you emails** - uses Twig
- **Echo** - uses custom MVC system
- **CentralNotice** - uses hacked up raw HTML / messages framework
- **Wikibase** - uses a custom template system
- **Wikia** - uses Nirvana on server-side, Mustache.js on client-side

and the list goes on...

Avoid jQuery spaghetti

Templating allows us to separate content/UI from code.

Building HTML in your code is messy and hard to read:

```
var locationDiv = $( '<div class="mwe-location mwe-upwiz-details-fieldname-input ui-helper-clearfix"></div>' )

    .append( $('<div class="mwe-location-label"></div>' ) )
    .append( mw.message( 'mwe-upwiz-location' ).escaped() )
    .addHint( 'location' ) )
    .append(
        $( '<div class="mwe-upwiz-details-input-error"><label class="mwe-validator-error" for="' + latId + '" generated="true"/></div>' ),
        $( '<div class="mwe-upwiz-details-input-error"><label class="mwe-validator-error" for="' + lonId + '" generated="true"/></div>' ),
        latDiv, lonDiv
    );
```

Minimum requirements

- Boolean logic (if/else)
- Array handling
- Secure (clear or automatic sanitization/escaping)
- Extensible enough to accommodate our i18n system
- No huge client-side footprint
- Readable
- Commentable

https://www.mediawiki.org/wiki/Requests_for_comment/HTML_templating_library#Requirements

Candidates

- Mustache/Hogan/Handlebars: PHP + JS
- Twig/Swig: PHP + JS
- Nirvana: PHP
- AngularJS: JS
- KnockoutJS: JS
- Write our own
- Combination of the above

Security

- Manual vs automatic context-sensitive escaping to avoid XSS
 - Security review & make no mistakes vs. let library do the work
 - Preferably automatic for messages and content (no review there)
 - Need DOM for automatic escaping, so excludes some candidates

Performance

Little difference betw.
PHP libraries

HHVM 1.4-2x
faster than PHP

Handlebars / Node.js
6-12x faster than
best HHVM

Engine	Test1	Test1b	Test2	Test3
Handlebars [Node.js]	0.118	0.158	0.589 / 1.235 (using lambda)	0.183
Mustache [Node.js]	0.339	0.384	2.393 / 3.720 (using lambda)	0.719
MediaWiki [HHVM]	0.922	0.970	8.803	4.402
Twig (string) [HHVM]	0.993	1.007	3.937	2.639
Mustache [HHVM]	1.159	1.200	5.539 / 12.696 (using lambda)	2.331
Twig (string) [PHP 5.3.10]	1.387	1.472	6.018	3.668
MediaWiki [PHP 5.3.10]	1.731	1.815	17.521	7.454
Mustache [PHP 5.3.10]	1.854	1.955	11.310 / 24.417 (using lambda)	3.728

Power

- logic-less vs. simple boolean logic / arithmetic expressions
 - For UI code: all minor template predicates in controller vs inline
 - For content / messages: need to expose JS vs. can get away with safe expressions for most use cases
- less powerful: easy to port, easy to read
- more power: possibly easier to use

Execution model

- Purely client-side app communicating with server via API vs. static server-side rendering
- or something in between: server-side pre-rendering with dynamic client-side updates / customization for logged-in users
- Affects need for sharing single solution between PHP and JS

Option 1: Long-term solution first

- Parsoid team to prototype simple DOM-based templating solution
 - can use existing JS code (KnockoutJS / AngularJS)
 - feature parity with Handlebars, PHP port
 - time-boxed: 3 months?
 - if fails, go with option 2

Option 2: Temporary solution first

- Pick existing string-based templating solution, for example Handlebars
 - Rely on manual sanitization & security review
 - Security regression vs. `Html::element`
- Develop longer-term DOM-based solution
- Replace string-based solution with safer version when ready